GMS: 공간 데이터베이스 관리 시스템

박상근*, 박순영*, 정원일*, 김명근*, 배해영*
* 인하대학교 전자계산공학과
{skpark, sunny, wnchung, kimmkeun}@dblab.inha.ac.kr, hybae@inha.ac.kr

GMS: Spatial Database Management System

Sang-Keun Park, Soon-Young Park, Warn-ill Chung, Myung-Keun Kim, Hae-Young Bae
Dept. of Computer Science & Engineering, Inha University

요 약

전통적인 관계형 데이터베이스 시스템에서 관리되고 있는 일반적인 데이터가 아닌 점, 선, 다각형 등의 다양한 공간 데이터를 관리하기 위해서는 확장된 형태의 공간데이터 타입 및 대용량성과 다양한 접근 패턴을 지니는 공간데이터의 특성을 고려한 새로운 데이터베이스 관리 시스템이 요구된다.

본 논문에서는 이와 같은 공간데이터의 특성을 고려한 저장 기법과 공간질의 처리 기법을 제공하는 공간 데이터베이스 관리 시스템인 GMS 를 제안한다. GMS 는 다양한 크기를 지니는 공간데이터의 특성을 고려하여 공간/비공간 통합 저장관리 및 BLOB 데이터 저장기법을 제공하며, 저장된 공간/비공간 데이터에 대한 다양한 색인기법을 제공하고 있다. 그 밖에 공간 연산 및 복잡한 질의처리를 위해 확장된 질의 최적화 및 질의처리 기법을 제공하며, 다중 사용자를 위한 확장된 동시성 제어 기법과 공간/비공간 데이터에 대한 서로 다른 회복 기법을 제공한다.

1. 서론

정보통신, 컴퓨터 시스템, 지도제작 및 자료처리 기술이 급속도로 발전하면서, 인간생활에 필요한 지리정보를 효율적으로 활용하기 위한 정보 시스템인 GIS(Geographical Information System)의 활용 분야는 환경의 개선, 효율적 토지정보의 관리에서부터 시설물관리, 국방정보관리, 자원관리, 도시계획 등에 이르기까지 적용 분야가 매우 넓고 다양해지게 되었다. 이렇듯 광범위한 GIS 적용분야와 다양한 환경 하의 공간데이터를 관리하기 위해서는 복잡한 공간 데이터의 저장과 관리, 대용량 공간 데이터의 저장, 다양한 형식의 공간 데이터에 대한 빠른 접근 방법, 다양한 공간 데이터 타입의 지원, 공간 객체간 연산 지원, 고비용의 공간연산 수행을 고려한 질의 처리 기법 등 공간 데이터의 특성을 고려한 저장관리 기법과 효율적 질의처리 기법을 제공하는 공간 데이터베이스 관리 시스템이 요구된다.

본 논문 [1]에서는 이와 같은 공간 데이터의 특성을 고려하여 설계, 구현한 공간 데이터베이스 관리 시스템인 GEOMania Millennium Server(이하 GMS)를 제안한다. GMS 는 Open-GIS 에 명시된 POINT, LINESTRING, POLYGON, MULTIPOLYGON, EOMETRYCOLLECTION 등의 다양한 공간데이터 타입 및 EQUALS, DISJOINT, INTERSECTS 등의 공간 연산을 위한 확장된 SQL, 수십 바이트에서 수 기가바이트에 이르는 다양한 크기의 공간데이터를 위한 공간/비공간 데이터의 통합 저장 관리 기법, 대용량의 공간데이터

[1] 본 연구는 대학 IT연구센터 육성.지원사업의 연구 결과로 수행되었음

래스터 맵, 멀티미디어 데이터 저장을 위한 BLOB 데이터 저장 기법을 제공하고 다양한 형식의 공간 데이터로의 빠른 접근을 위한 공간색인, 복잡한 질의와 공간 연산에 대한 확장된 질의 최적화 기법, 다중 사용자를 위한 확장된 동시성 제어 기법, 공간/비공간 데이터를 구분한 확장된 회복 관리 기법을 제공한다.

본 논문의 구성은 다음과 같다. 2 장에서는 기존의 대용량 데이터 저장 기법과, 연쇄적 질의 처리를 위한 대수 연산자에 대해 설명하고 3 장에서는 GMS 시스템 구조에 대해 간략히 설명하며 4 장에서는 대용량 데이터 및 공간/비공간 속성의 저장을 고려한 GMS 데이터베이스 구조와 로깅(logging) 데이터의 안정성 및 다중 사용자를 위한 확장된 저장관리 기법에 대해 설명한다. 5 장에서는 복잡한 질의와 공간 질의 최적화 및 디스크 접근빈도를 고려한 질의처리 기법에 대해 설명하며, 마지막으로 6 장에서 결론을 맺기로 한다.

2. 관련연구

2.1 기존 대용량 데이터 저장 기법

저장 시스템의 관점에서, 데이터의 저장을 위해 필요한 공간이 한 페이지 이상인 경우를 대용량 데이터라고 한다[12]. 기존의 저장 시스템은 일반적으로 한 페이지보다 작은 텍스트나, 수치 데이터만을 고려하여 설계되었기 때문에, 대용량 데이터를 효율적으로 처리하지 못하는 문제점이 있다. 이런 문제점을 해결하기 위해 Informix, Sybase 등의 상용 데이터베이스 관리 시스템은 대용량 데이터를 연결된 리스트의 형태로 저장하는 BLOB 관리 기법을 사용하고 있다. 그 외에 EXODUS 는 대용량 데이터에 대해 B^+-Tree 구조의 색인을 이용하여 임의의 접근이 가능하도록 했고, Starburst 는 실제 데이터가 저장되는 세그먼트에 대한 개수 및 크기 정보를 지니는 롱 필드 서술자(long field descriptor)를 이용한 버디 시스템을 사용하고 있다. 그러나 이러한 기법들은 순차접근과 임의접근 모두를 만족하지는 못하며, 대용량 데이터의 경우만을 고려하였기 때문에 수십 바이트에서 수 기가에 이르는 다양한 공간데이터의 저장구조에 적용할 경우 저장공간 낭비 및 이로 인한 빈번한 디스크 입/출력 때문에 시스템의 성능이 저하된다는 단점이 있다[12].

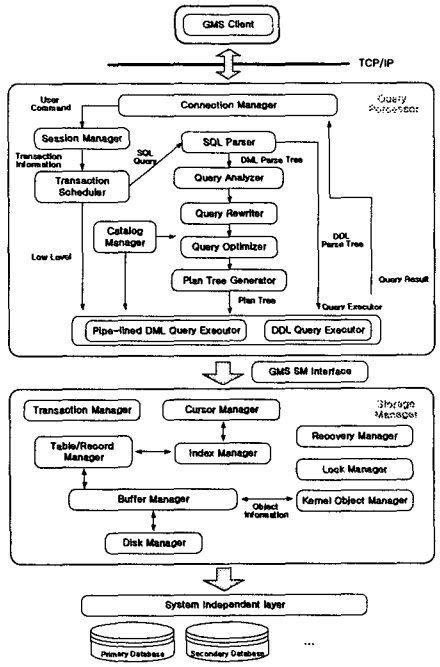
2.2 연쇄적 질의 처리를 위한 대수 연산자

대수 연산자의 구현 기법은 일반적으로 임시 릴레이션(Temporary Relation)을 이용한 실체화(Materialization)기법과 반복자 (Iterator)를 이용한 연쇄적 처리(Pipe-lined Processing) 기법 두 가지로 나뉜다[9,10]. 전자는 수행단위가 각 대수 연산자에 입력되는 튜플의 집합이며, 연산 후에 결과의 집합을 임시 릴레이션으로 저장하여 다음 연산의 입력값으로 사용한다. 이는 구현 알고리즘이 간단한 대신 임시 릴레이션에 대한 디스크 공간과 그에 따른 접근 비용이 증가하기 때문에 대용량 데이터베이스에 적용할 경우 성능 저하를 초래하게 된다. 후자의 경우 수행단위가 각 대수 연산자에 입력되는 튜플이며(*Tuple-at-a-time*), *Group* By나 Order By의 집합 연산을 제외하고는 임시 릴레이션을 생성하지 않고 *Open*(), GetNext(), Colse()의 세 가지 인터페이스를 사용하여 질의를 연쇄적으로 수행한다. 각각의 대수 노드는 일정크기의 입/출력 버퍼를 가지며, 하위 노드의 출력버퍼는 그대로 상위 노드의 입력 버퍼가 되며, 각 노드에 대한 병렬적 수행이 가능하게 된다. 이로 인해 각 노드의 수행 시간이 중첩되어 질의 처리에 걸리는 전체 시간을 감소시키며, 멀티프로세스와 많은 디스크를 가진 서버기종에서 최대한의 성능을 발휘할 수 있도록 한다

3. GMS 시스템 구조

GMS 시스템 구조는 크게 질의 처리기와 저장 관리자로 나눌 수 있으며

GMS 시스템 구조도는 <그림 3-1>과 같다.



<그림 3-1> GMS 시스템 구조

GMS 시스템 구조 중 세션관리기는 일반 사용자 레벨 및 고급 관리자 레벨의 세션을 따로 유지함으로써 비정상적인 사용자 세션 종료로부터 시스템의 리소스를 보호하고, 트랜잭션을 관리하는 역할을 수행한다. SQL 파서(Parser)는 기본적인 SQL 외에 Open-GIS 에서 명시된 공간 질의를 위한 확장된 SQL 을 지원한다. 질의 분석기(Query Analyzer)는 파싱(Parsing) 후 전달된 질의에 대한 질의 분석 과정을 통해, 문법적 오류 여부를 확인하고 질의 처리를 위한 정보를 질의 블록에 작성한다. 질의 재작성기(Query Rewriter)는 질의 분석 과정을 통해 작성된 부질의에 대한 질의 블록이 있는 경우, 이를 부질의에 부합되는 테이블의 질의 블록과 통합한다. 질의 최적화기(Query Optimizer)와 플랜트리 작성기(Plan Tree Generator)는 테이블에 대한 통계정보를 이용해 기본 플랜트리를 구성한 후 CNF(Conjunction Normal Form)으로 구성된 필터(filter)정보를 플랜트리 내 작성하고 플랜트리 노드의 실제 수행방법을 지정한다. 질의 수행기(Query Executor)는 질의 최적화기로부터 전달된 플랜트리에 대한 질의처리를 수행한다. 트랜젝션(Transaction) 관리기는 질의에 대한 허용범위 제어를 위해 5 단계 독립성 레벨(Isolation Level)을 지원하고 있다. 커서(Cursor) 관리기는 데이터의 접근비용을 줄이기 위해 레코드의 연속적 접근을 지원한다. 색인(Index) 관리기는 공간/비공간 데이터에 대한 빠른 접근을 위해 확장된 Disk Based Extendible Hashing, B^+-Tree, R-Tree[7] 색인을 제공한다. 객체(Object) 관리기는 GMS 에서 객체로 취급되는 테이블과 인덱스의 생성, 삭제 및 빠른 접근을 위한 구조를 제공한다. 레코드(Record) 관리기는 공간/비공간 데이터 및 BLOB 데이터의 저장과 고비용 연산 시 디스크 접근을 고려한 구조를 제공한다. 버퍼(Buffer) 관리기는 Steal, No Force 정책을 사용하고 있으며, 버퍼의 사용빈도 분산을 위해 버퍼 풀을 그룹(group) 별로 나누어 관리한다. 디스크(Disk) 관리기는 대용량 데이터를 지원하기 위해 64 비트의 주소를 이용, 페이지에 접근하고 있으며 데이터베이스 내 하나의 세그먼트는 최대 4 테라(Tera) 크기의 데이터를 저장할 수 있다. 잠금(Lock) 관리기는 데이터의 효율적 접근을 고려한 다단위 락(mulit-granularity locking) 기법을 제공한다. 회복(Recovery) 관리기는 ARIES 회복 알고리즘[2]을 기반으로 하며, 공간/비공간 데이터의 회복을 고려한 확장된 회복 기법을 제공하고 있다.

4. GMS 저장 관리자

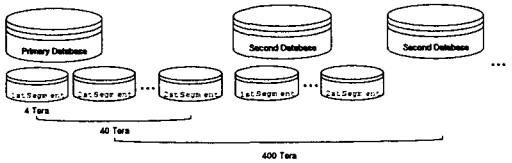
4.1 GMS 데이터베이스 구조

GMS 는 공간/비공간 데이터의 통합 저장 및 대용량 공간/멀티미디어 데이터의 저장을 고려한 대용량 저장구조를 지니고 있다.

4.1.1 전체 데이터베이스 구조

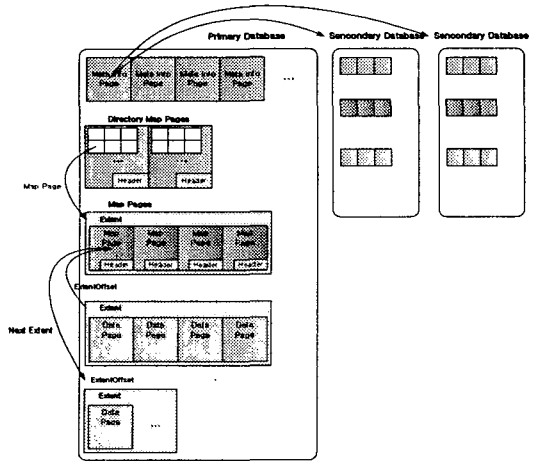
GMS 는 최대 10 개의 데이터베이스와

하나의 데이터베이스 내 10 개의 세그먼트를 유지할 수 있으며, 한 세그먼트 당 4 테라(Tera)의 데이터를 저장할 수 있어 총 400 테라에 달하는 방대한 양의 데이터를 저장할 수 있다. <그림 4-1>은 GMS 데이터베이스 구조를 나타낸다.



<그림 4-1> GMS 데이터베이스 구조

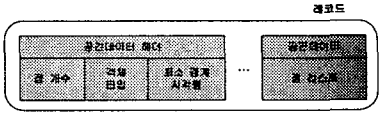
GMS 의 세그먼트는 O/S 파일 시스템의 특성과 디스크 접근 비용을 고려 파일 입출력 단위를 페이지 단위로 구분하며, 페이지들은 인접한 데이터의 접근성을 고려하여 익스텐트라는 4 개의 페이지 묶음으로 관리되고 있다. 익스텐트는 디스크 영역의 할당단위가 된다. GMS 는 이와 같은 대용량의 데이터 구조를 관리하기 위해 [그림 4-2]와 같은 구조를 이용, 페이지, 인스텐트 등의 데이터 저장구조에 대한 빠른 접근경로를 제공하는 익스텐트 맵, 파일 맵 등의 맵 페이지와 맵 페이지의 접근을 위한 디렉토리 맵 페이지 구조를 이용한 다단계 관리기법을 사용하고 있다.



<그림 4-2> 대용량 데이터의 관리를 위한 구조

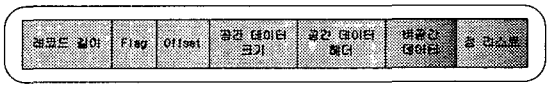
4.1.2 공간/비공간 속성 및 대용량 객체 저장을 위한 레코드 구조

GMS 의 레코드는 복잡한 구조와 다양한 크기를 가지는 공간 데이터의 특성을 고려하여 <그림 4-3>과 같은 구조를 이용함으로써 정적 공간 데이터 헤더를 통해 공간연산을 위한 여과과정이나 공간색인(R-Tree) 구성 시 발생되는 고비용의 재구성 비용을 줄이고 비정형적인 실제 공간 데이터 부분을 나누어 관리함으로써 저장공간에 대한 활용도를 높일 수 있다.

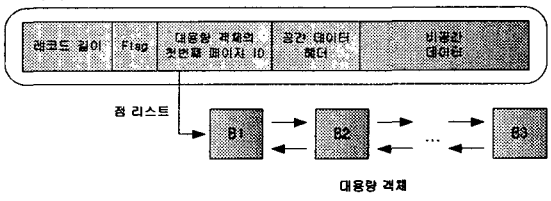


<그림 4-3> 공간 데이터 구조

또한, 대용량성이라는 공간데이터의 특성을 고려 레코드의 크기가 특정 크기 이하이면 [그림 4-4]의 (a)와 같은 일반 레코드 구조로 저장하고 특정크기 이상인 경우 [그림 4-4]의 (b)와 같은 대용량 객체 타입으로 저장한다. 대용량 객체의 경우 일반 데이터 페이지에는 속성 레코드와 공간 데이터에 대한 헤더 정보만을 저장하고 실제 공간 데이터는 대용량 객체 페이지에 저장함으로써 다양한 크기의 공간데이터에 대한 저장공간 이용률을 높일 수 있다[12].



(a)공간 데이터의 크기가 작은 경우



(b)공간 데이터의 크기가 클 경우

<그림 4-4>GMS 레코드 구조

4.2 메모리 사용빈도 분산을 위한 메모리 관리 구조

데이터베이스는 디스크 입출력의 기본

단위인 페이지에 대한 입력과 출력을 위해 접근되며 모든 연산은 메모리에서 수행하게 된다. GMS 는 이를 위해 주 기억 장치 공간을 페이지 단위로 분할하여 관리하게 되는데 이를 버퍼(buffer)라 하며, 버퍼들의 연속적인 집합을 버퍼 풀(buffer pool)이라 한다[11]. GMS 는 효율적인 접근 성능에 비해 용량의 제한이 따르는 버퍼 풀을 효과적으로 관리하기 위한 버퍼교체정책으로 페이지의 중요성에 가중치를 두는 참조 계수를 적용한 LRU(Least Recently Used) 정책을 사용한다. LRU 정책의 장점으로는 성능/구현 비가 좋다는 점을 들 수 있다. 그 외에 GMS 는 버퍼에 대한 사용과 관리의 분산을 위해 버퍼에 대한 데이터 pointer, 메모리 상의 버퍼주소, 버퍼의 latch 상태 등 버퍼에 대한 정보를 지니는 버퍼 아이템을 그룹별로 나누어 관리하고 있다. 이는 각 그룹에 대해 독립성을 부여하는 효과를 가져오며 독립된 그룹은 버퍼에 대한 latch 의 사용이 원활하게 되어 보다 효율적인 동시성 제어를 제공하게 된다.

4.3 다중 사용자를 위한 확장된 동시성 제어 기법

GMS 는 다중사용자에 대한 데이터베이스의 일관성을 유지하기 위한 잠금 기법으로 S2PL(Strict Two-Phase Locking)기법 사용하고 있다. S2PL 은 어떤 트랜잭션 T 가 기록해 놓은 값을 T 가 철회되거나 완결되기 전까지 다른 트랜잭션들이 판독하거나 덮어쓰지 않도록 하는 기법을 말한다[11]. 이러한 S2PL 의 스케줄은 cascade rollback 을 방지하고, 수정된 객체들의 원래 값을 복원함으로써 철회 트랜잭션을 시행 할 수 있다는 장점을 지닌다. GMS 는 이러한 S2PL 기법과 더불어 다른 객체를 포함하고 있는 객체에 대해 효율적으로 락를 설정 할 수 있도록 함으로써 락 획득에 드는 비용을 최소화 시키는 다단계 락킹규약(multi-graularity locking protocol)을 제공한다. 또한 하부 객체의 락 획득수가 락 한계치에 도달하면 락 단위를 상향 조정함으로써 락 오버헤드를 줄이기 위한 락 상승(lock escalation)기법을 제공한다. 드물게 락 상승기법은 잠금을 상승 시킬 경우 상승된 잠금 모드와 다른 잠금 모드간에 충돌을 야기시키기도 하는데 GMS 는 이러한 경우 락 상승을 하지 않고 락 카운트만 증가시킴으로써 락 상승으로 인한 DeadLock 의 발생 가능성을 배제시킨다. 이 외에도 GMS 는 다중사용자에 대한 병행성을 높이기 위해 트랜잭션의 특성 중 하나인 독립성 레벨을 Dirty write, dirty read, read committed, repeatable read, serializable 의 5 단계로 나누고 있다. 일반적인 4 종류의 단계외에 Dirty write 레벨이 추가되었는데 이 레벨은 변경이 거의 일어나지 않으며 변경에 대한 영향이 중요하지 않은 메타정보에 적용되고 있다. 이 같은 독립성 레벨은 동시에 실행되고 있는 다른 트랜잭션의 작업들에 대해 주어진 트랜잭션이 노출되는 정도를 제어하게 되는데 5 단계의 독립성 레벨 중 하나를 선택함으로써 사용자는 다른 트랜잭션들의 미완결 변경에 대해 현행 트랜잭션의 노출도를 높이는 대신 더 높은 병행성을 얻을 수 있다는 특징을 지니게 된다. [표 4-1]은 트랜잭션 독립성 레벨에 따른 효과를 보여준다.

적용성 레벨	오손판독	반복 불가능 판독	변동
Dirty Write	가능성 있음	가능성 있음	가능성 있음
Dirty Read	가능성 있음	가능성 있음	가능성 있음
Read Committed	없음	가능성 있음	가능성 있음
Repeatable Read	없음	없음	가능성 있음
Serializable	없음	없음	없음

<표 4-1> 트랜잭션 독립성 레벨에 따른 효과

4.4 로깅(logging) 데이터의 안정성 및 대용량 공간 데이터의 회복을 고려한 회복 관리 기법

GMS 의 회복기법은 steal, no force 정책의 ARIES 회복 기법을 기반으로 하고 있다. 그러나, 페이지의 반환이 발생할 경우에는 트랜잭션의 지연을 위해

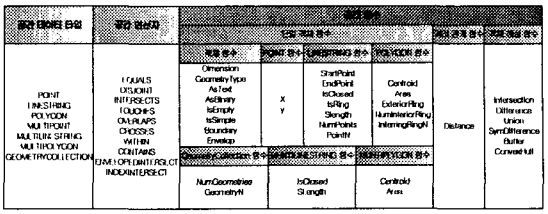
pending list 라는[2] 자료구조를 이용하며, pending list 에는 각각의 트랜잭션들에 대해서 트랜잭션의 완료시점까지 지연시켜야 할 연산을 가지고 있게 된다. pending list 를 이용해 트랜잭션을 지연시키는 것은 삭제 과정에서 반환된 페이지를 다른 트랜잭션이 획득함으로써 발생하게 되는 before image 에 대한 유실 현상으로 인해 트랜잭션의 취소가 불이행되는 것을 막기 위함이다. 그 외에 로깅 데이터의 안정성을 위해 로깅된 데이터를 복수로 유지함으로써 회복 파일의 미디어 실패로 인해 발생하는 시스템 실패로부터 시스템을 정상적으로 복구하기 위한 기법이 연구중이다.

5. GMS 질의 처리기

GMS 질의처리기는 공간 연산을 위한 확장된 SQL 을 지원하며 복잡한 질의와 공간 연산의 최적화 외에 고비용 연산과정에서 발생하는 디스크 접근 비용를 최소화 하기위한 확장된 질의 처리 기법을 사용한다.

5.1 기본 SQL 지원 및 공간 연산을 위한 확장된 SQL

GMS 는 전통적 데이터베이스 관리 시스템에서 다루는 일반적인 데이터 외에 공간 데이터에 대한 질의처리를 위해 확장된 GMS SQL parser 를 제공하고 있다. GMS SQL parser 는 비공간 질의를 위해 SQL-92 를 지원하며 공간 질의를 위해서는 Open-GIS 에서 명시된 공간 데이터 타입 및 공간 질의 문법을 지원하고 있다. 또한 공간 테이블 생성을 위해 확장된 DDL(Data Definition Language)와 바이너리 및 BLOB 데이터를 정의하기 위해 확장된 DML(Data Manipulation Language)를 지원한다. [표 5-1]은 GMS SQL parser 가 지원하는 확장된 SQL 중 공간데이터 타입, 공간 연산, 공간 함수에 관한 목록을 보여주고 있다.

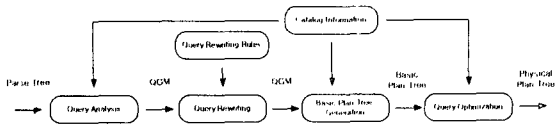


<표 5-1> 공간데이터를 위한 질의 확장

5.2 다중 테이블과 복잡한 서브 질의에 대한 연산 및 공간 연산 최적화를 위한 기법

테이블간의 조인을 갖는 질의나 부 질의를 갖는 질의 등 복잡한 질의는 테이블에 대한 조인 수행 순서나 조인 연산을 위한 카티션 프로덕트(cartesian product)의 방향, 필터 조건의 적용 시기 등에 따라 현저한 성능차이를 보인다. GMS 는 이와 같은 최적화 요소를 활용하여 최소의 수행비용으로 질의를 처리하기 위해, 질의 처리 초기에 생성된 주 질의 및 부 질의들에 대한 질의 처리 정보를 담고 있는 질의 블록을 통합하고, 인덱스의 사용가능 여부를 확인하며 카타로그 테이블이 지니고 있는 통계정보를 이용한 block based nested loop join 에 대한 loop 내 위치를 결정한 후, 조건절에 사용된 테이블 정보를 표현한 DAG(Directed Acyclic Graph) 기법을 통하여 최적의 질의 플랜을 작성한다. GMS 질의 최적화기는 실질적인 질의 수행을 위한 최소비용 플랜트리를 생성하기 위해 먼저 조인연산에 참여하는 테이블간의 순서를 결정하는 기본 플랜 트리(Basic Plan Tree)를 작성하며, 이후 기본 트리의 노드들인 테이블들에 대한 조건절을 사용하여 최적화된 논리적 플랜트리(Logical Plan Tree)를 작성하고, 마지막으로 테이블로부터 조건에 부합하는 레코드들에 대한 접근 경로를 포함하는 물리적 플랜 트리(Physical Plan Tree)를 작성한다. 또한 GMS 질의 최적화기는 비공간 질의와 달리 공간 질의 최적화를 위해 공간 연산에 대한 최적화 경로를 후방향으로 하는 확장된 질의 최적화 기법을 사용하고 있다. <그림 5-1>는 이와

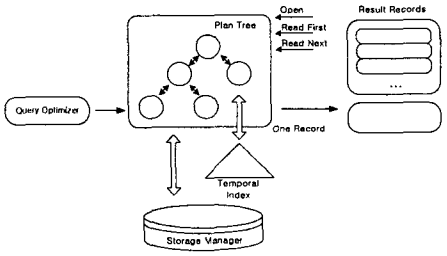
같은 질의 최적화 수행 단계를 나타낸다.



<그림 5-1> 질의 최적화 수행 단계

5.3 디스크 접근 비용을 고려한 질의 처리 기법

GMS 는 비공간 DML 처리를 위해 Sort-merge 조인, Hash 조인 등과 같은 기존의 조인 기법과 임시 인덱스(temporary index)를 이용한 Filter push down 조인 기법을 제공하고 있다. GMS 는 디스크 접근 비용을 고려하여 기존의 임시테이블을 이용하는 방법 대신, 모든 질의수행을 반복자 인터페이스 내에서 하나의 레코드씩 처리하는 Pipe-lined Processing 기법을 사용한다. 이와 같은 질의 처리 기법은 질의 수행 시 임시테이블을 만드는 비용을 제거하며 사용자에게 응답시간을 단축시키는 장점을 지닌다. 그 외에 distinct 질의처리나 group by 를 이용한 집계 함수 질의 처리 시에는 디렉토리 분할에 따른 공간 낭비를 줄인, 확장된 Disk Based Extendible Hashing 기법을 사용한다.



<그림 5-2> 질의 처리 전략

<그림 5-2>는 Pipe-lined Processing 과정에서 임시 인덱스를 이용한 Filter push down 조인 기법을 수행하는 실행 관리기의 수행단계를 나타내고 있다.

6. 결론 및 향후 연구방향

본 논문에서 제안한 GMS 는 복잡, 다양한 형태와 대용량성 이라는 공간데이터의 특성을 고려하여 데이터의 효율적 저장, 관리 및 빠를 질의처리 수행을 위해 개발한 공간 데이터베이스 관리 시스템이다. GMS 저장 관리자는 대용량 데이터베이스 관리를 위해 다단계 관리기법을 사용하며 효율적 동시성 제어를 위해 버퍼의 그룹별 관리, 확장된 다단위 락 기법, 5 단계의 트랜잭션 독립성 레벨을 지원한다. 그 외에 공간/비공간 데이터에 대한 서로 다른 회복 기법을 제공하여 회복 비용을 최소화 하였다. GMS 질의 처리기는 공간 질의에 대한 최적화를 위해 최적화 경로를 후방향으로 수행하며 고비용의 공간 연산을 고려해 디스크 접근 비용을 최소화 한 Pipe-lined Processing 기법을 사용하고 있다.

현재 진행중인 연구로는 GMS 를 이용한 고확장, 고가용성 클러스터 데이터 베이스 관리 시스템인 GMS Cluster 가 개발 마무리 단계에 있으며, 능동적 coordinator 를 이용한 분산 데이터베이스 관리 시스템인 GMS* 와 GMS 를 이용한 LBS system 이 개발 중에 있다.

참고문헌

[1] Guting, R., "An Introduction to Spatial Database Systems", VLDB Journal, Vol.3, pp. 357-399, 1994.

[2] C. Mohan, et al., "ARIES : A Transaction Recovery Method Supporting Fine-Granularity Locking and Partial Rollbacks Using Write-Ahead Logging," IBM Research Report RJ6649, IBM Almaden Research Center, 1989.

[3] EXODUS Project Group, "Using the EXODUS Storage System V3.1" EXODUS Project Document, University of Wisconsin-Madison, 1993

[4] Transaction Support in an Extensible Operating System. Yasushi Saito and Brian N. Bershad. 1998 USENIX Annual Technical Conference. New Orleans, LA. June 1998.

[5] M. J. Garey, et al, "Object and File Management in the EXODUS Extensible Database System," Proc. Of 12th VLDB,

pp.357-383, 1986.

[6] Mohan, C., Narang, I., "Recovery and Coherency-Control Protocols for Fast Intersystem Page Transfer and Fine-Granularity Locking in a Shared Disks Transaction Environment," Proc. 17th International Conference on Very Large Data Bases, Barcelona, September 1991.

[7] Guttman, A., "R-Trees: An Dynamic Index Structure for Spatial Searching," proc. Of ACM SIGMOD Int'l Conf. On Management of Data, pp.47-57, 1984.

[8] J. Gray and A. Reuter, "Isolation Concepts," in Transaction Processing: Concepts and Techniques, San Mateo, CA, Morgan Kaufmann Publishers, 1993, pp. 403-406.

[9] Hector Garcia-Molina, Jeffry D Ulman, Jennifer Widom, "Database System Implementation," Prentice Hall, 2000.

[10] Laura M. Haas, J. C. Freytag, G. M. Lohman, H. Pirahrsh, "Extensible Query Processing in Starburst," ACM SIGMOD Record, Proc. Of the 1989 ACM SIGMOD international conference on Management of data, Vol. 18, Issue 2, pp. 377-388, June 1989.

[11] Ramakrishnan, J. Gehrke, "Database Management Systems", McGraw Hill College Div, 2002.

[12] 김대일, "공간저장관리장의 공간데이타 특성을 고려한 저장관리 기법," 인하대학교 전자계산공학과 대학원 석사학위논문, 2000.

[13] 김종현, "KORED/StormNT : 공간 데이터베이스를 위한 저장관리자의 설계 및 구현," 한국 정보과학회 춘계학술대회 발표 논문집, Vol. 28, No. 1, 2001.

[14] 김홍연, "공간 데이터베이스 시스템을 위한 고비용 술어 질의 최적화," 공학 박사 학위 논문, 인하대학교, 1999.